Programmation Avancée et Compilation 2001
(Exercises)

2 Lexical Analysis

Keywords 2

Give short explanations of the following keywords:

> *Token*, *terminal*, *nonterminal*, *BNF*, *EBNF*, *regular expression*, *regular language*.

Exercise 2

Consider the following two grammars producing expressions of the kinds Aexpr and Bexpr:

1.	Aexpr	::=	"–" | Digit | Aexpr Digit
2.	Bexpr	::=	Digit Bexpr | Bexpr Digit | "."
	Digit	::=	"0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

(a) Describe the languages produced by the two grammars.
Hint: Derive some words first so to get acquainted with the grammars.

(b) Both grammars are *recursive*, that is, the left-hand nonterminals appear on the right-hand sides of some of the productions. Give equivalent grammars using non-recursive EBNF notation.

Exercise 3 (Programming exercise)

In http://lampwww.epfl.ch/courses/progavancee01/HandScanner.java, we have provided a little scanner which recognizes tokens for the following grammar:

```
LPAREN ::= "("
RPAREN ::= ")"
PLUS   ::= "+"
MINUS  ::= "–"
TIMES  ::= "*"
DIV    ::= "/"
INTLIT ::= ("0" | "1" | "2" | "3"| "4" | "5" |"6"| "7" | "8" | "9")
           { ("0" | "1" | "2" | "3"| "4" | "5" |"6"| "7" | "8" | "9") }
```

Unfortunately, the nextToken() function is missing. It is your task to write it.

The function does not have to care about positions, but you have to store the input-characters in the variable chars. You can set chars to the empty string at the beginning and then append further characters with chars += (**char**) ch. To read the input, the implementation already provides a function nextCh() which yields –1 when EOF is reached.

The class HandScanner has a main routine which tests the scanner by reading from the standard input and calling nextToken() repeatedly. An example input is provided in http://lampwww.epfl.ch/courses/progavancee01/example.expr.